COMPSYS 723: EMBEDDED SYSTEM DESIGN

Nilesh Magan and Kathryn Jaggar

Department of Electrical and Computer Engineering
University of Auckland, Auckland, New Zealand

Abstract

In this report, we present the design of a cruise controller in Esterel. The cruise controller is a safety critical system making it an excellent candidate for use of this synchronous programming language. We first describe the specification and present the according Finite State Machine. We discuss a multi-module Esterel design and present the various interfaces. Finally, we explain our approach to testing this system.

1. Introduction

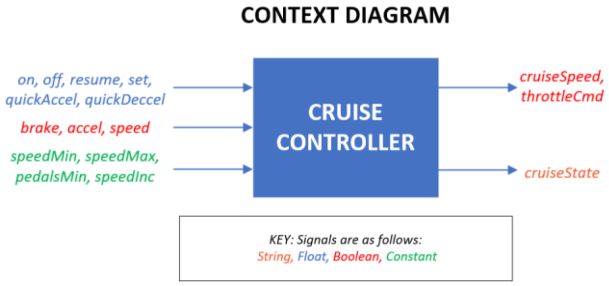
The cruise controller controls the cruise speed of the vehicle based on the speed input requests from the user, braking or accelerating actions of the user and the predefined speed limits (maximum and minimum). Control operations include regulating cruise speed at the set value, increase cruise speed when the quick accelerate button is pressed, and visa versa for the quick deceleration button press. In this report, we will first specify the design and then present our Esterel mapping of this specification. We will illustrate the use of several of the Esterel language features which have allowed concurrency and synchronisation. We also illustrate the use of data handling functions in C, the use of multiple modules, the connection of ports and of interfaces.

The organization of this report is as follows. Section 2 presents the specification of the cruise controller. In section 3 we present the Esterel design, including the interfaces, top-level module and causality. In section 4 we present our approach to testing and finally, in section 5, we present our conclusion.

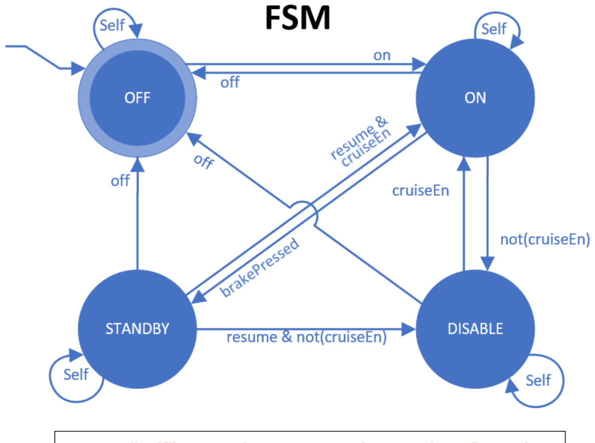
2. Specification

Described in figure 1 is the overall input-output interface of the system. User input actions include a number of buttons and accelerator/ brake pedal actions. The remaining input is the vehicles current speed state. The cruise controller reads the various inputs and determines which cruising state should be entered.

Outputs of the system include cruise speed, throttle command and the current cruise state. These are also presented in figure 1. The cruise speed represents the current speed the vehicle is traveling at, the throttle command represents a percentage of how much fuel is supplied to the engine, and the cruise state represents the which of the four cruising states.


CONTEXT DIAGRAM
on, off, resume, set, quickAccel, quickDeccel
brake, accel, speed
speedMin, speedMax, pedalsMin, speedInc
CRUISE CONTROLLER
cruiseSpeed, throttleCmd
cruiseState
KEY: Signals are as follows:
String, Float, Boolean, Constant

The Finite State Machine (FSM) in figure 2 presents the four states (Off, On, Standby and Disable) as well as the conditions that must be met in order to reach each state. In the Off state the cruise controller is not enabled and the vehicle speed is not regulated. Thus it is driven by the user's input directly. If the on button is pressed then the system moves to the On state, but all other button presses have no effect. Once in the On state the cruise controller is enabled and the cruise speed will be regulated. This speed is set to the value of the input speed when the on button was pressed. So long as the accelerator or brake pedal is not pressed the controller will remain in the On state. If the accelerator is pressed, or the cruise speed is outside of the predefined maximum/minimum speed limit, then the controller transitions to the Disable state. In this state the cruise speed will remain constant while the throttle command will be driven by the accelerator input. The controller will return to the On state when the accelerator is no longer being pressed and the speed is within the defined limits. Similarly, the controller will transition from the On state to the Standby state if the brake pedal is pressed. In this state the cruise speed will also remain constant and the throttle command will be driven by the accelerator pedal. A transition from this state requires a press of the resume button and the destination will be determined based on if the accelerator is pressed and if the speed is within the defined limits. The transition will either be to the On state or to the Disable state. In all states a press of the off button will cause a transition to the Off state.



NOTE: "Self" Is just the transition that is taken if no other inputs trigger other transitions

In any of the On, Standby or Disable states the set button may be pressed to update the cruise speed to the speed input, and the quick accelerate/decelerate buttons will cause an increase or decrease of the cruise speed by a predefined amount when pressed.

3. Design in Esterel

The Esterel design of the cruise controller follows directly from the previous specification. There are six modules making up the Esterel program, and one top-level named cruiseControl.

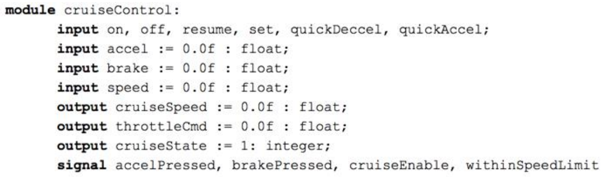
- Module 1, FSM: This modules executes the FSM previously described which transitions from one state to the next depending on various inputs and internal signals received from other modules. The FSM is depicted in figure 2. The FSM is emulated using state variables and traps to emulate go-tos.
- Module 2, brakeDetection: This module performs a simple comparison of values to determine if the brake pedal has been pressed. The brake input must exceed a predefined percentage value in order for the brake pressed to be registered.
- Module 3, accelDetection: Similar to brakeDetection this modules performs a simple comparison of values to determine if the acceleration pedal has been pressed. The accel input must exceed a predefined percentage value in order for the acceleration pressed to be registered.
- Module 4, speedLimitDetection: This module performs a simple value comparison between the speed input and the minimum and maximum speed limit values. It determines if the current speed is within the predefined speed limits.
- Module 5, cruiseEnableDetection: This module determines if the current state of inputs allow for the cruise state to be enabled (ie not disabled). This is determined based on whether the accelerator is pressed, and the vehicle is within the speed limits. This is determined using the present command.
- Module 6, cruiseSpeedManagement: This module handles updating the cruise speed appropriately depending on various inputs and internal singles. The cruise speed must be set when either the on or set buttons are pressed, to a value within the speed limit. Thus the cruise speed may be set to the speed input or saturated to either of the maximum and minimum speed limit values. Similarly, the cruise speed should be updated by the predefined amount when either the quickAccel or quickDecel buttons are pressed. Again the cruise speed should be set to either the speed input +/- the predefined amount or saturated to the maximum and minimum speed limits respectively.

Each of the above-explained modules execute as concurrent threads as defined in the top level cruiseControl module. The that module also performs the port mapping of the remaining six modules can be seen in Appendice A at the end of the document.

3.1. Interfaces

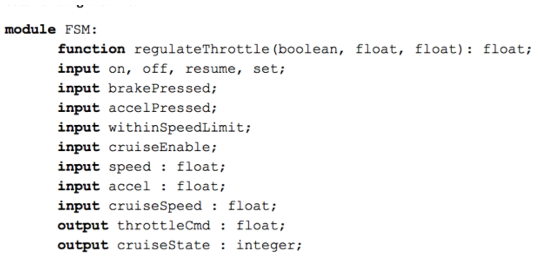
The interface of a module includes data type declarations, constants, functions, input and output signals. A visual representation of the module interfaces is depicted in figure. Our design includes the following interfaces;

The interface of the top level module includes the system inputs and outputs, in addition to internal singles used throughout the design by other modules.



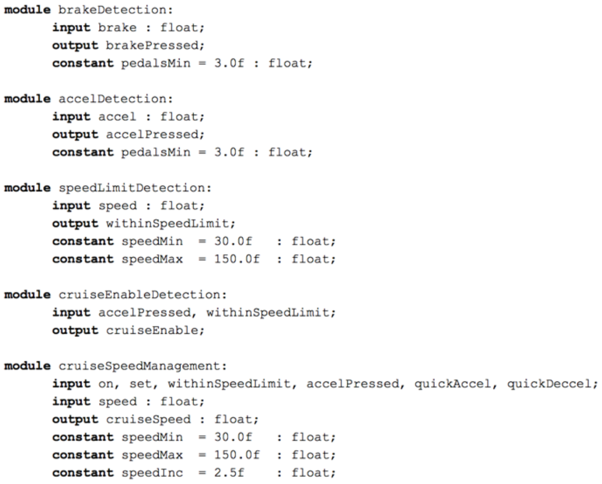
```
module cruiseControl:
        input on, off, resume, set, quickDeccel, quickAccel;
        input accel := 0.0f : float;
        input brake := 0.0f : float;
        input speed := 0.0f : float;
        output cruiseSpeed := 0.0f : float;
        output throttleCmd := 0.0f : float;
        output cruiseState := 1: integer;
        signal accelPressed, brakePressed, cruiseEnable, withinSpeedLimit
```

Note here that the internal signals defined above are now inputs or outputs of other various modules amongst the system inputs and outputs. The interface of the FSM module below also includes a function which allows for data handling within C.



```
module FSM:
      function regulateThrottle(boolean, float, float): float;
      input on, off, resume, set;
      input brakePressed;
      input accelPressed;
      input withinSpeedLimit;
      input cruiseEnable;
      input speed : float;
      input accel : float;
      input cruiseSpeed : float;
      output throttleCmd : float;
      output cruiseState : integer;
```

The following module interfaces include constant values in addition to inputs and outputs. These constants are predefined and used as reference points for comparisons.



```
module brakeDetection:
      input brake : float;
      output brakePressed;
      constant pedalsMin = 3.0f : float;

module accelDetection:
      input accel : float;
      output accelPressed;
      constant pedalsMin = 3.0f : float;

module speedLimitDetection:
      input speed : float;
      output withinSpeedLimit;
      constant speedMin  = 30.0f   : float;
      constant speedMax  = 150.0f  : float;

module cruiseEnableDetection:
      input accelPressed, withinSpeedLimit;
      output cruiseEnable;

module cruiseSpeedManagement:
      input on, set, withinSpeedLimit, accelPressed, quickAccel, quickDeccel;
      input speed : float;
      output cruiseSpeed : float;
      constant speedMin  = 30.0f   : float;
      constant speedMax  = 150.0f  : float;
      constant speedInc  = 2.5f    : float;
```

3.2. The Top Level Module

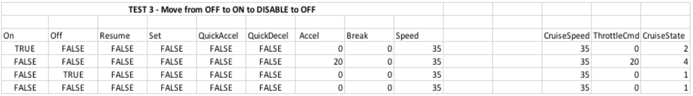
The purpose of the top level module cruiseControl is to run the remaining six modules in parallel. This required interconnecting the interface ports correctly such that the output of one module becomes another modules input and visa versa. In order to use the signal renaming approach, all inputs, outputs, and singles must be declared in the top level interface, as seen previously.

3.3. Causality

When sharing signals between modules it is important to ensure the composition is still causal. We have achieved this in a number of places through the use of Esterel’s pre command. This command uses the previous value/state of the signal ensuring causal cycles do not exist. An example of this can be found in the CruiseSpeedManagement module. When emitting the cruiseSpeed value often we must know the cruiseSpeed value, and in these instances, we use the pre command.

4. Testing

A number of test cases were developed in order to ensure this safety critical system was functionally correct. Our approach to testing involved the development of testing input and output vectors in an excel document such that we could manually test using the Esterel GUI. Through this testing, we were able to find a number of bugs. Due to the short nature of tests and the use of the GUI tree window we were able to easily find the cause of our errors.



TEST 3 - Move from OFF to ON to DISABLE to OFF

On	Off	Resume	Set	QuickAccel	QuickDecel	Accel	Break	Speed			CruiseSpeed	ThrottleCmd	CruiseState
TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	0	0	35			35	0	2
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	20	0	35			35	20	4
FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	0	0	35			35	0	1
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0	0	35			35	0	1

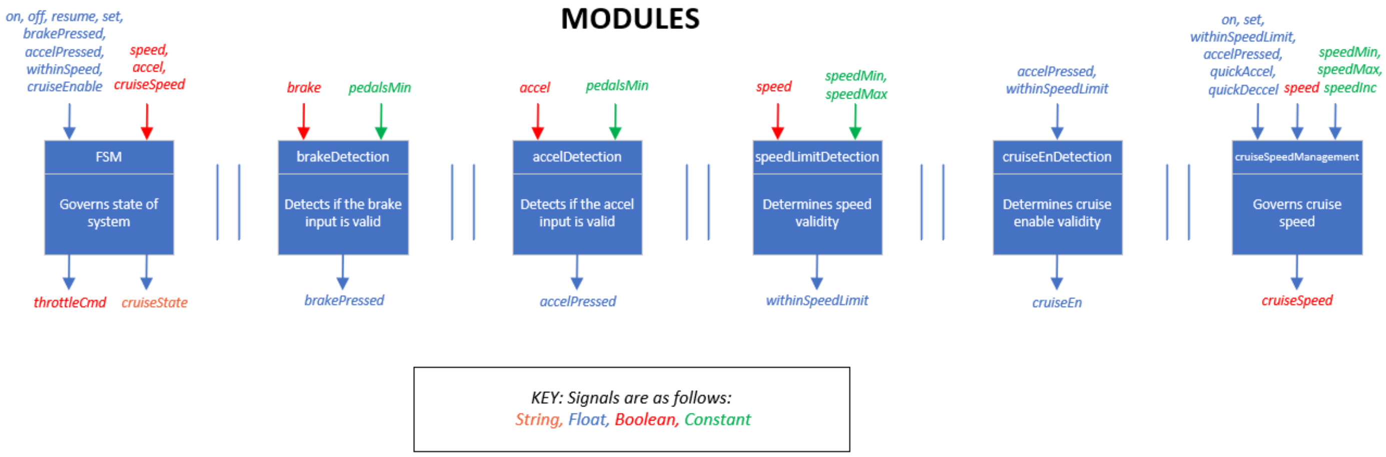
In total 15 tests were created, which can be found in the appendix. One example is depicted in figure 4. Each test includes a test number and short description of what is being tested. Inputs are displayed on the left and expected outputs on the right. Values for each tick are stated on a new row.

5. Conclusions

We have demonstrated the design capabilities of the Esterel language by implementing a small safety-critical design. We illustrate many Esterel capabilities, concurrency, synchrony and data handling using C. We can conclude that Esterel is a suitable tool used to implement this system and similar due to the ease in which we can achieve concurrency. Although we undertook extensive manual testing we would suggest that formal verification methods were used to ensure the system is functionally correct if the design were to be used for an application.

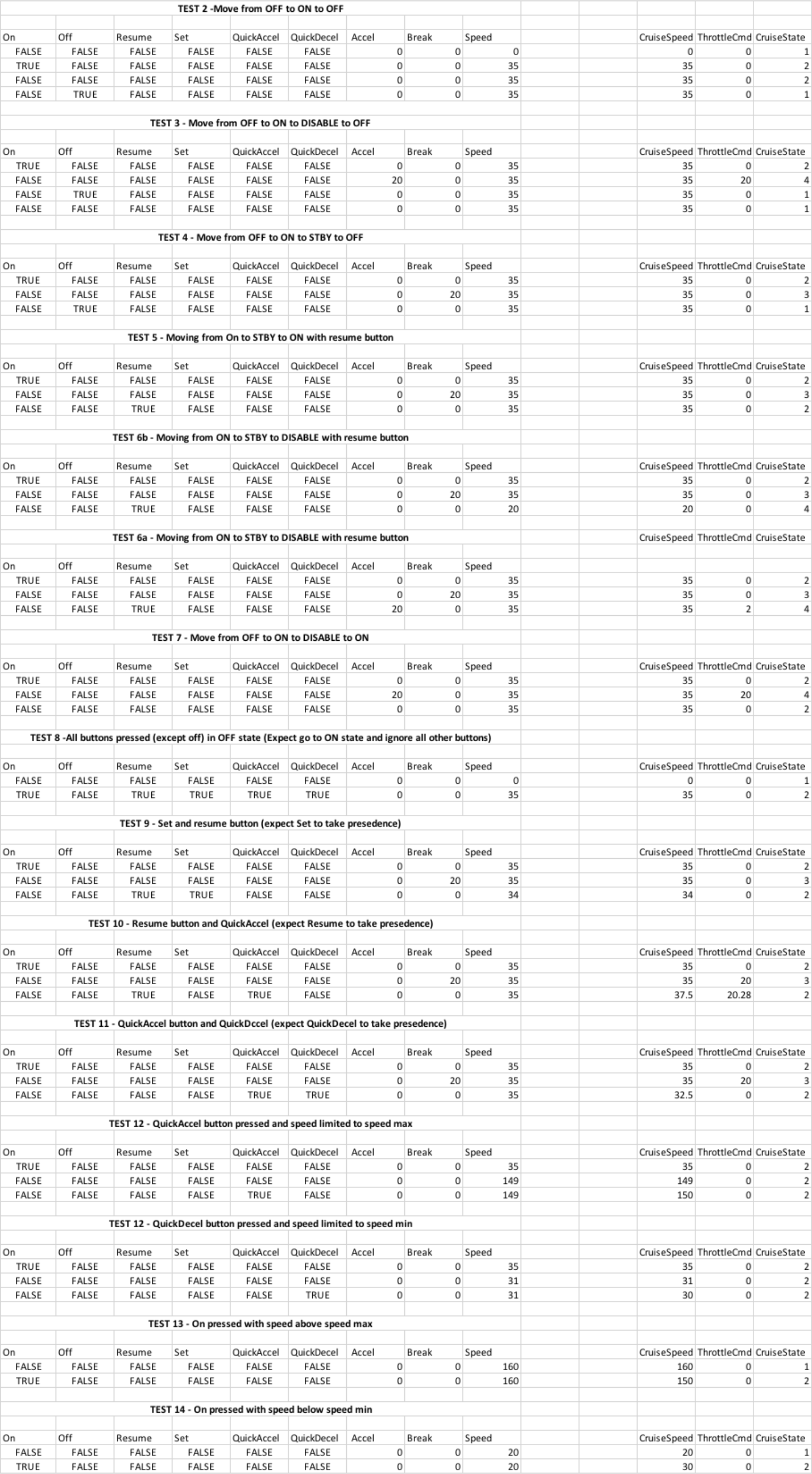
Appendix

Appendice A


MODULES
on, off, resume, set, brakePressed, accelPressed, withinSpeed, cruiseEnable
speed, accel, cruiseSpeed
FSM
Governs state of system
throttleCmd
cruiseState
brake
pedalsMin
brakeDetection
Detects if the brake input is valid
brakePressed
accel
pedalsMin
accelDetection
Detects if the accel input is valid
accelPressed
speed
speedMin, speedMax
speedLimitDetection
Determines speed validity
withinSpeedLimit
accelPressed, withinSpeedLimit
cruiseEnDetection
Determines cruise enable validity
cruiseEn
on, set, withinSpeedLimit, accelPressed, quickAccel, quickDeccel
speed
speedMin, speedMax, speedInc
cruiseSpeedManagement
Governs cruise speed
cruiseSpeed
KEY: Signals are as follows:
String, Float, Boolean, Constant

Appendice B is continued on the next page…

Appendice B



TEST 2 -Move from OFF to ON to OFF

On	Off	Resume	Set	QuickAccel	QuickDecel	Accel	Break	Speed	CruiseSpeed	ThrottleCmd	CruiseState
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0	0	0	0	0	1
TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	0	0	35	35	0	2
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0	0	35	35	0	2
FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	0	0	35	35	0	1

TEST 3 - Move from OFF to ON to DISABLE to OFF

On	Off	Resume	Set	QuickAccel	QuickDecel	Accel	Break	Speed	CruiseSpeed	ThrottleCmd	CruiseState
TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	0	0	35	35	0	2
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	20	0	35	35	20	4
FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	0	0	35	35	0	1
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0	0	35	35	0	1

TEST 4 - Move from OFF to ON to STBY to OFF

On	Off	Resume	Set	QuickAccel	QuickDecel	Accel	Break	Speed	CruiseSpeed	ThrottleCmd	CruiseState
TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	0	0	35	35	0	2
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0	20	35	35	0	3
FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	0	0	35	35	0	1

TEST 5 - Moving from On to STBY to ON with resume button

On	Off	Resume	Set	QuickAccel	QuickDecel	Accel	Break	Speed	CruiseSpeed	ThrottleCmd	CruiseState
TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	0	0	35	35	0	2
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0	20	35	35	0	3
FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	0	0	35	35	0	2

TEST 6b - Moving from ON to STBY to DISABLE with resume button

On	Off	Resume	Set	QuickAccel	QuickDecel	Accel	Break	Speed	CruiseSpeed	ThrottleCmd	CruiseState
TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	0	0	35	35	0	2
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0	20	35	35	0	3
FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	0	0	20	20	0	4

TEST 6a - Moving from ON to STBY to DISABLE with resume button

On	Off	Resume	Set	QuickAccel	QuickDecel	Accel	Break	Speed	CruiseSpeed	ThrottleCmd	CruiseState
TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	0	0	35	35	0	2
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0	20	35	35	0	3
FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	20	0	35	35	2	4

TEST 7 - Move from OFF to ON to DISABLE to ON

On	Off	Resume	Set	QuickAccel	QuickDecel	Accel	Break	Speed	CruiseSpeed	ThrottleCmd	CruiseState
TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	0	0	35	35	0	2
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	20	0	35	35	20	4
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0	0	35	35	0	2

TEST 8 -All buttons pressed (except off) in OFF state (Expect go to ON state and ignore all other buttons)

On	Off	Resume	Set	QuickAccel	QuickDecel	Accel	Break	Speed	CruiseSpeed	ThrottleCmd	CruiseState
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0	0	0	0	0	1
TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	0	0	35	35	0	2

TEST 9 - Set and resume button (expect Set to take presedence)

On	Off	Resume	Set	QuickAccel	QuickDecel	Accel	Break	Speed	CruiseSpeed	ThrottleCmd	CruiseState
TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	0	0	35	35	0	2
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0	20	35	35	0	3
FALSE	FALSE	TRUE	TRUE	FALSE	FALSE	0	0	34	34	0	2

TEST 10 - Resume button and QuickAccel (expect Resume to take presedence)

On	Off	Resume	Set	QuickAccel	QuickDecel	Accel	Break	Speed	CruiseSpeed	ThrottleCmd	CruiseState
TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	0	0	35	35	0	2
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0	20	35	35	20	3
FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	0	0	35	37.5	20.28	2

TEST 11 - QuickAccel button and QuickDccel (expect QuickDecel to take presedence)

On	Off	Resume	Set	QuickAccel	QuickDecel	Accel	Break	Speed	CruiseSpeed	ThrottleCmd	CruiseState
TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	0	0	35	35	0	2
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0	20	35	35	20	3
FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	0	0	35	32.5	0	2

TEST 12 - QuickAccel button pressed and speed limited to speed max

On	Off	Resume	Set	QuickAccel	QuickDecel	Accel	Break	Speed	CruiseSpeed	ThrottleCmd	CruiseState
TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	0	0	35	35	0	2
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0	0	149	149	0	2
FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	0	0	149	150	0	2

TEST 12 - QuickDecel button pressed and speed limited to speed min

On	Off	Resume	Set	QuickAccel	QuickDecel	Accel	Break	Speed	CruiseSpeed	ThrottleCmd	CruiseState
TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	0	0	35	35	0	2
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0	0	31	31	0	2
FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	0	0	31	30	0	2

TEST 13 - On pressed with speed above speed max

On	Off	Resume	Set	QuickAccel	QuickDecel	Accel	Break	Speed	CruiseSpeed	ThrottleCmd	CruiseState
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0	0	160	160	0	1
TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	0	0	160	150	0	2

TEST 14 - On pressed with speed below speed min

On	Off	Resume	Set	QuickAccel	QuickDecel	Accel	Break	Speed	CruiseSpeed	ThrottleCmd	CruiseState
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0	0	20	20	0	1
TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	0	0	20	30	0	2